Technical Disclosure Commons

Defensive Publications Series

July 2023

Application Testing Under Developer Specified Device Resource Occupancy

Tracy Wang

Wendy Dai

Follow this and additional works at: https://www.tdcommons.org/dpubs_series

Recommended Citation
Wang, Tracy and Dai, Wendy, "Application Testing Under Developer Specified Device Resource Occupancy", Technical Disclosure Commons, (July 18, 2023)
https://www.tdcommons.org/dpubs_series/6065

This work is licensed under a Creative Commons Attribution 4.0 License.
This Article is brought to you for free and open access by Technical Disclosure Commons. It has been accepted for inclusion in Defensive Publications Series by an authorized administrator of Technical Disclosure Commons.

Application Testing Under Developer Specified Device Resource Occupancy

ABSTRACT

During normal usage, consumer devices may remain switched on without a shutdown and restart for long durations of time. A lengthy period of time since the last restart can lead to high usage of device resources such as CPU, memory, storage, etc. Program performance issues as well as errors caused by these are hard to detect using clean functional test environments. This disclosure describes techniques to emulate end-user scenarios as lengthy times since last restart and high resource utilization by providing the developer with the ability to easily configure the usage of the CPU, memory, and storage of a device-under-test (DUT) via a device resources management tool. The device resources management tool is implemented such that it can invoke low level operating system APIs to occupy a specified percentage of resources such as CPU, memory, storage, etc. The extent to which each device resource is occupied can be set in an independent or combined manner. The device resources management tool enables developers to emulate various real world resource utilization scenarios and can help identify bugs that are otherwise rare and/or difficult to reproduce.

KEYWORDS

- Device under test (DUT)
- Resource utilization
- Resource usage
- Resource occupancy
- Application testing
- Functional testing
- Device restart

BACKGROUND

During normal usage, consumer devices such as smartphones, tablets, smart watches, laptops, etc. may remain switched on without a shutdown and restart for long durations of time. A lengthy period of time since the last restart can lead to high usage of device resources such as CPU, memory, storage, etc. Program performance issues as well as errors may also be caused by this. However, functional test environments in the factory or hardware/software development laboratory are generally clean, e.g., devices under test (DUT) are flashed, factory-reset, or set up from scratch. While clean device resets are intended to ensure isolated test environments, e.g., no interference from previous test runs, bugs that surface at times of high resource usage and/or upon substantial duration of time since last restart are difficult to detect.

Applications that can monitor and display the usage of device resources are available. However, these apps do not occupy resources to simulate end-user scenarios for application testing. Longevity testing can validate product stability and serviceability over long periods under load and stress conditions that emulate real time traffic and applications. Bugs that arise from lengthy times since restart or from high resource usage can be detected by longevity testing. However, by definition, longevity tests take long hours to run and to learn the results. Longevity tests are also expensive due to needing support of test infrastructure, monitoring effort, and additional code to simulate data traffic.

DESCRIPTION

This disclosure describes techniques to emulate such end-user scenarios as lengthy times since last restart and high resource usage by providing developers with the ability to easily configure the usage of the CPU, memory, and storage of the device under test (DUT). A device resources management tool, e.g., in the form of a lightweight archive file such as application

package file (a file that includes all the data an app needs, including code, assets, and resources) invokes low level operating system application programming interfaces (APIs) to occupy a specified percentage of CPU, memory, storage, etc. The extent to which each device resource is occupied can be set independently and on a per-request basis. The device resources management tool serves as a device resources management tool that can also reset CPU, memory, and storage resources separately.

The device resources management tool runs on the device under test (DUT). It can use standard programming libraries (supported by the device operating system) and can override dynamic memory allocation during runtime. The device resources management tool can be exposed to the user via a command line interface (CLI) that enables the developer to communicate with the device. The device resources management tool can be invoked from the command line interface for manual testing/checking. Alternatively, commands to the device resources management tool can be incorporated into testing code for automated testing and development. During execution, the device resources management tool can monitor resource usage and maintain it at or above the threshold level specified by the developer.

Command	Arguments	Description	Range for the argument
occupy()	cpu_tasks	Number of running tasks to occupy CPU	Integer between 0-100
	memory_size	Occupied memory in percentage	Number between 0-100
	storage_size	Occupied storage size in percentage	Number between 0-100
reset()	cpu, memory, storage, all (default)	Reset one or more resources	String enum

Fig. 1: Example commands to occupy device resources at specified levels and to reset resource occupancy

Fig. 1 illustrates examples of syntax for commands to occupy device resources at specified levels and to reset resource occupancy. Under the specifications of Fig. 1, a command such as `occupy –cpu_tasks 6 –memory_size 90` results in the CPU being occupied by six tasks and the memory being occupied to the extent of 90%. Similarly, a command such as `reset all` results in the device resource occupancy to be reset.

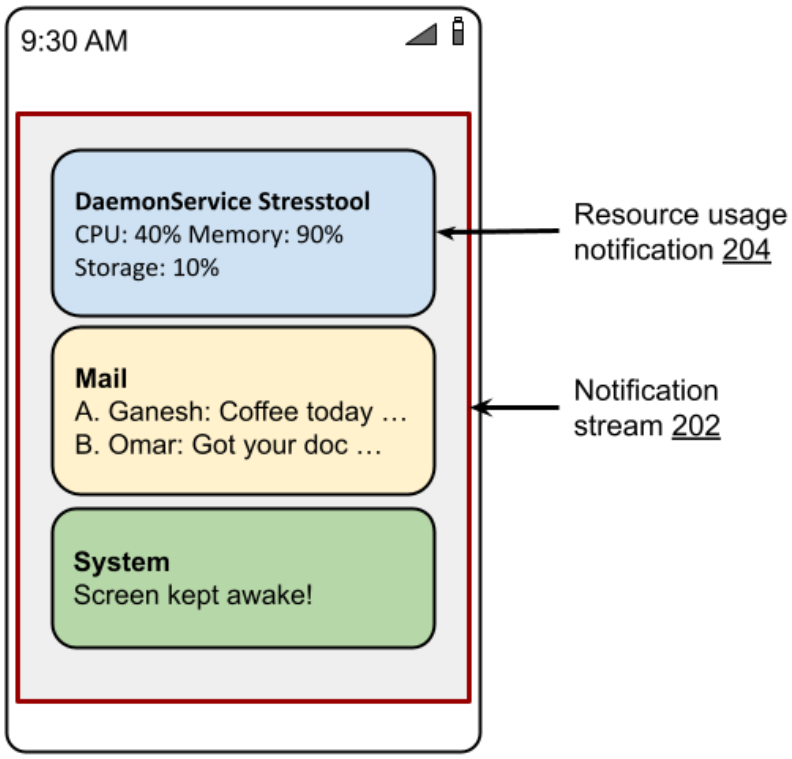

Fig. 2: Output of the device resources management tool can be integrated with the device notification stream

As illustrated in Fig. 2, output of the device resources management tool can be integrated with notifications (202) generated by the device operating system. After a user sets the usage levels and the tool starts occupying resources accordingly, a notification (204) is popped up, which the user can tap to check the occupancy of the CPU, memory, or storage. The notification can be updated periodically.

The described device resources management tool enables developers to specify parameters of resource usage for a device, thereby enabling easy emulation of end-user device scenarios (of high resource utilization). The device resources management tool can enable developers to improve productivity and reduce turnaround time in many ways. It can assist in defect detection during the early development stage. It can significantly shorten the time for troubleshooting and reproducing a bug that can take weeks or months of real-world scenario to be detected. The described techniques are scalable and applicable to a wide range of consumer devices such as smartphones, smartwatches, tablets, cameras, etc. and for any operating system.

CONCLUSION

This disclosure describes techniques to emulate end-user scenarios as lengthy times since last restart and high resource utilization by providing the developer with the ability to easily configure the usage of the CPU, memory, and storage of a device-under-test (DUT) via a device resources management tool. The device resources management tool is implemented such that it can invoke low level operating system APIs to occupy a specified percentage of resources such as CPU, memory, storage, etc. The extent to which each device resource is occupied can be set in an independent or combined manner. The device resources management tool enables developers to emulate various real world resource utilization scenarios and can help identify bugs that are otherwise rare and/or difficult to reproduce.

REFERENCES

[1] “Device Info: System Info & CPU - Apps on Google Play” available online at https://play.google.com/store/apps/details?id=com.hardwarecheck.sensorcheck accessed June 26, 2023.

[2] “Castro Premium - system info - Apps on Google Play” available online at https://play.google.com/store/apps/details?id=com.itemstudio.castro.pro&hl=en&gl=US accessed June 26, 2023.